Red Hat JBoss Core Services 2.4.57

Apache HTTP Server Connectors and Load Balancing Guide

For Use with Red Hat JBoss Core Services 2.4.57

Last Updated: 2024-03-15

Red Hat JBoss Core Services 2.4.57 Apache HTTP Server Connectors and Load Balancing Guide

For Use with Red Hat JBoss Core Services 2.4.57

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Install and configure load-balancing solutions that use the mod_jk and mod_proxy_cluster HTTP connectors along with other modules that Red Hat JBoss Core Services provides.

Table of Contents

PROVIDING FEEDBACK ON RED HAT JBOSS CORE SERVICES DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to **create a ticket**.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. HTTP CONNECTORS

Red Hat JBoss Core Services (JBCS) includes two different HTTP connectors that the Apache HTTP Server can use to load-balance HTTP requests to a set of back-end servlet containers:

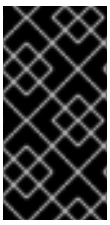
- The Apache Tomcat connector (**mod_jk**) supports the load balancing of HTTP requests to a set of servlet containers, while maintaining sticky sessions and communicating over the Apache JServ Protocol (AJP).
- The JBoss HTTP connector (**mod_proxy_cluster**) is a more advanced load balancer than **mod_jk**. The **mod_proxy_cluster** connector provides all the functionality of **mod_jk** and additional features such as real-time load-balancing calculations, application life-cycle control, automatic proxy discovery, and multiple protocol support.

JBCS and Red Hat Enterprise Linux (RHEL) provide separate distributions of the Apache HTTP Server. You can use the JBCS distribution of the Apache HTTP Server to connect to back-end application servers by using the **mod_jk**, **mod_proxy_cluster**, or **mod_proxy** connector as a proxy.

Consider the following guidelines:

- From RHEL 7 onward, the JBCS and RHEL distributions of the Apache HTTP Server provide identical **mod_proxy** modules.
- On RHEL versions 7 and 8, only the JBCS distribution of the Apache HTTP Server provides the **mod_jk** and **mod_proxy_cluster** connectors.
- From RHEL 9 onward, the JBCS and RHEL distributions of the Apache HTTP Server also provide identical copies of the **mod_jk** connector and the **mod_proxy_cluster** connector.
- Installing a JBCS distribution of the Apache HTTP Server from an archive file or from RPM packages by using the **groupinstall** option also automatically installs the **mod_jk** and **mod_proxy_cluster** connectors.
- Installing the RHEL 9 distribution of the Apache HTTP Server does not automatically install the **mod_jk** and **mod_proxy_cluster** connectors. In this situation, you can manually install the appropriate **mod_jk** or **mod_proxy_cluster** package by using RHEL Application Streams. For more information, see **Mod_jk** installation and **Mod_proxy_cluster** installation and upgrade.

The Apache HTTP Server Connectors and Load Balancing Guide describes how to install and configure the **mod_jk** and **mod_proxy_cluster** connectors that JBCS provides. This guide also includes a working example for basic load-balancing with mod_proxy_cluster.



IMPORTANT

Most file and directory paths shown in this guide are for an archive installation of JBCS on Red Hat Enterprise Linux. For other platforms, use the correct paths for your respective installation, as specified in the Red Hat JBoss Core Services Apache HTTP Server Installation Guide.

Additional resources

- Load balancing with the Apache Tomcat connector (**mod_jk**)
- Load balancing with the JBoss HTTP connector (**mod_proxy_cluster**)

CHAPTER 2. LOAD BALANCING WITH THE APACHE TOMCAT CONNECTOR (MOD_JK)

The Apache Tomcat Connector, **mod_jk**, is a plug-in that allows the Apache HTTP Server to forward web requests to a back-end servlet container. The **mod_jk** module also allows the Apache HTTP Server to load-balance requests to a set of servlet containers, while maintaining sticky sessions.

2.1. MOD_JK INSTALLATION

Red Hat JBoss Core Services (JBCS) and Red Hat Enterprise Linux (RHEL) provide separate distributions of the Apache HTTP Server. The Apache HTTP Server distribution that you install determines whether installation of the **mod_jk** connector is automatic or requires a manual step. Depending on your installed distribution of the Apache HTTP Server, the installation path for the **mod_jk** module and configuration files also varies.

NOTE

The JBCS Apache HTTP Server supports the use of **mod_jk** on all supported operating systems. The RHEL Apache HTTP Server supports the use of **mod_jk** on RHEL 9 only.

2.1.1. Installation of mod_jk when using the JBCS Apache HTTP Server

The Apache HTTP Server part of a JBCS installation automatically installs the **mod_jk** module.

You can follow the procedures in the Red Hat JBoss Core Services Apache HTTP Server Installation Guide to install the JBCS Apache HTTP Server for your operating system. For more information, see the **Additional resources** links.

Consider the following guidelines for a **mod_jk** installation when using the JBCS Apache HTTP Server:

- The **mod_jk.so** module is installed in the ***JBCS_HOME*/httpd/modules** directory.
- The **mod_jk.conf.sample**, **workers.properties.sample**, and **urworkermap.properties.sample** configuration files are located in the ***JBCS_HOME*/httpd/conf.d** directory.
- The **mod_jk.conf.sample** file includes a **LoadModule** directive for the **mod_jk** module.

NOTE

JBCS_HOME represents the top-level directory for a JBCS installation, which is **/opt/jbcs-httpd24-2.4**.

Additional resources

- Installing the JBCS Apache HTTP Server on RHEL from archive files
- Installing the JBCS Apache HTTP Server on RHEL 7 or RHEL 8 from RPM packages
- Installing the JBCS Apache HTTP Server on Windows Server

2.1.2. Installing mod_jk by using RHEL Application Streams

If you install the RHEL 9 distribution of the Apache HTTP Server from an RPM package by using Application Streams, RHEL does not automatically install the **mod_jk** package. In this situation, if you want to use the **mod_jk** connector, you must install the **mod_jk** package manually.

Prerequisites

- You have installed the Apache HTTP Server on RHEL 9 by using Application Streams.

Procedure

- Enter the following command as the root user:

```
# dnf install mod_jk
```

Verification

- To check that the **mod_jk** package is successfully installed, enter the following command:

```
# rpm -q mod_jk
```

  The preceding command outputs the full name of the installed package, which includes version and platform information.

Consider the following guidelines for a **mod_jk** installation when using RHEL Application Streams:

- The **mod_jk.so** module is installed in the **/usr/lib64/httpd/modules** directory.
- The **mod_jk.conf.sample**, **workers.properties.sample**, and **urworkermap.properties.sample** configuration files are located in the **/etc/httpd/conf.d** directory.
- The **mod_jk.conf.sample** file includes a **LoadModule** directive for the **mod_jk** module.

Additional resources

- Application Streams
- Managing software with the DNF tool

2.2. APACHE HTTP SERVER LOAD-BALANCING CONFIGURATION WHEN USING MOD_JK

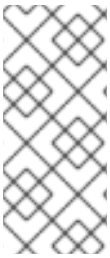
You can configure the Apache HTTP Server to use the **mod_jk** connector to load-balance requests to a set of servlet containers. This setup includes the configuration of back-end worker nodes.

Depending on whether you installed **mod_jk** through Red Hat JBoss Core Services (JBCS) or by using Red Hat Enterprise Linux (RHEL) Application Streams, consider the following guidelines:

- JBCS provides example configuration files for **mod_jk** in the ***JBCS_HOME*/httpd/conf.d/** directory.
- RHEL provides example configuration files for **mod_jk** in the **/etc/httpd/conf.d/** directory.

The example configuration files for **mod_jk** are named **mod_jk.conf.sample**, **workers.properties.sample**, and **uriworkermap.properties.sample**. To use these examples instead of

creating your own configuration files, you can remove the **.sample** extension, and modify the file content as needed.



NOTE

You can also use the Load Balancer Configuration tool on the Red Hat Customer Portal to generate optimal configuration templates quickly for **mod_jk** and Tomcat worker nodes. When you use the Load Balancer Configuration tool for Apache HTTP Server 2.4.57, ensure that you select **2.4.x** as the Apache version, and select **Tomcat/JWS** as the back-end configuration.

NOTE

Red Hat JBoss Core Services 2.4.57 does not support the tunneling of non-upgraded connections to a back-end WebSockets server. This means that when you are configuring the **ProxyPass** directive for the **mod_proxy_wstunnel** module, you must ensure that the upgrade parameter is not set to **NONE**. For more information about **mod_proxy_wstunnel**, see the Apache documentation.

2.2.1. Configuring the Apache HTTP Server to load mod_jk

You can configure the Apache HTTP Server to load **mod_jk**, by specifying configuration settings in the **mod_jk.conf** file. Depending on the Apache HTTP Server distribution that you are using, the location of the configuration file varies.

You can also perform the following optional configuration steps:

- In addition to the **JkMount** directive, you can use the **JkMountFile** directive to specify the configuration file for a mount point. The configuration file contains multiple URL mappings for Tomcat forwarding.
- You can configure the Apache HTTP Server that is functioning as the load balancer to log details of each worker node that handles a request. This can be useful if you need to troubleshoot your load balancer.

Prerequisites

- You have installed the Apache HTTP Server.
- If you installed the RHEL distribution of the Apache HTTP Server by using Application Streams, you have installed **mod_jk** manually.

Procedure

1. Go to the Apache HTTP Server configuration directory:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.
2. Create a new file named **mod_jk.conf** and enter the following configuration details:

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
```

```
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf.d/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSL KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Require ip 127.0.0.1
</Location>
```

IMPORTANT

Ensure that the **LoadModule** directive references the **mod_jk** native binary that you have installed.

NOTE

The **JkMount** directive specifies the URLs that the Apache HTTP Server can forward to the **mod_jk** module. Based on the configuration for the **JkMount** directive, **mod_jk** forwards the received URL to the correct servlet containers.

To enable the Apache HTTP Server to serve static content (or PHP content) directly, and only use the load balancer for Java applications, the preceding configuration example specifies that the Apache HTTP Server sends only requests with the URL /**application**/* to the **mod_jk** load balancer.

Alternatively, you can configure the Apache HTTP Server to forward all URLs to **mod_jk** by specifying /* in the **JkMount** directive.

3. Optional: To use the **JkMountFile** directive to specify the configuration file for a mount point, perform the following steps:

   a. Go to the Apache HTTP Server configuration directory:

      - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
      - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.

   b. Create a file named **uriworkermap.properties**.

   c. Specify the URL that you want to forward and the worker name.
   For example:

```
# Simple worker configuration file

# Mount the Servlet context to the ajp13 worker
/application=loadbalancer
/application/*=loadbalancer
```

NOTE

The required syntax is in the format: /***URL=WORKER_NAME***

The preceding example configures **mod_jk** to forward requests for **/application** to the JBoss Web Server Tomcat back end.

   d. In the **mod_jk.conf** file, enter the following directive:

```
# Use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf.d/uriworkermap.properties
```

4. Optional: To enable Apache HTTP Server logging, perform either of the following steps:

   - Include **%w** in your **JkRequestLogFormat** directive, as shown in the preceding step about **mod_jk.conf** settings.
   - Log the name of the **mod_jk** worker that you want to use, by including **%{JK_WORKER_NAME}n** in your Apache HTTP Server **LogFormat**(s).

Additional resources

- The Apache Tomcat Connectors - Web Server HowTo : mod_jk Directives
- Apache HTTP Server Documentation: Log Files

2.2.2. Configuring worker nodes in mod_jk

You can configure multiple worker nodes to handle the requests that the Apache HTTP Server forwards to the servlet containers, by specifying settings in the **workers.properties** file. Depending on the Apache HTTP Server distribution that you are using, the location of the configuration file varies.

The example in this procedure shows how to define two **mod_jk** worker nodes in a weighted round-robin configuration that uses sticky sessions between two servlet containers.

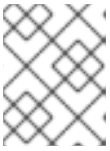
Prerequisites

- You are familiar with the format of the **workers.properties** directives.
- You have configured the Apache HTTP Server to load **mod_jk**.

Procedure

1. Go to the Apache HTTP Server configuration directory:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.
2. Create a file named **workers.properties**.
3. Enter the following configuration details:

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1
worker.node1.secret=<YourSecret>

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1
worker.node1.secret=<YourSecret>

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```



NOTE

In the preceding example, ensure that you replace **host**, **port**, and **secret** settings with values that are relevant for your environment.

IMPORTANT

The **secret** property is required when using the Tomcat AJP Connector. You can specify the **secret** property for a worker node or a load balancer in the **workers.properties** file. For example:

worker.*<WORKER_NAME>*.secret=*<YOUR_AJP_SECRET>*

In the preceding example, replace ***<WORKER_NAME>*** and ***<YOUR_AJP_SECRET>*** with values that are relevant for your environment.

2.2.3. Configuring JBoss Web Server to work with mod_jk

By default, JBoss Web Server is configured to receive Apache JServ Protocol (AJP) traffic from the **mod_jk** connector. On the JBoss Web Server host, the AJP connector is configured by default in the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.

However, to use a worker node with **mod_jk**, you must perform the following additional configuration steps:

- On the JBoss Web Server host, in the **server.xml** file, you must configure a unique value for the **jvmRoute** attribute in the Engine of each worker node.
- On the Apache HTTP Server host, in the **workers.properties** file, you must specify the **secret** property for a worker node or a load balancer. Depending on the Apache HTTP Server distribution that you are using, the location of the **workers.properties** file varies.

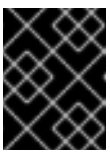
NOTE

The **secret** property is required when you use the Tomcat AJP connector.

Procedure

1. On the JBoss Web Server host, to configure a unique value for the **jvmRoute** attribute in the Engine of each worker node:
   a. Open ***JWS_HOME*/tomcat_<VERSION>_/conf/server.xml** file.
   b. Enter the following details:

```
<Engine name="Catalina" jvmRoute="node1" >
```



IMPORTANT

Ensure that the **jvmRoute** attribute value matches the worker name that you specify in the **workers.properties** file on the Apache HTTP Server host.

2. On the Apache HTTP Server host, to specify the **secret** property for a worker node or a load balancer:

a. Go to the Apache HTTP Server configuration directory:

   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.

b. Open the **workers.properties** file.

c. Ensure that the **secret** property is specified in the following format:

```
worker.<WORKER_NAME>.secret=<YOUR_AJP_SECRET>`
```

NOTE

Ensure that you replace ***<WORKER_NAME>*** and ***<YOUR_AJP_SECRET>*** with values that are appropriate for your environment.

NOTE

If you set a **secret** on a load balancer by using the **ProxyPass** directive, all members of the load balancer inherit this **secret**. For example:

```
<Proxy balancer://mycluster>`
   BalancerMember ajp://node1:8009 route=node1
secret=YOUR_AJP_SECRET
   BalancerMember ajp://node2:8009 route=node2
secret=YOUR_AJP_SECRET
</Proxy>
ProxyPass /example/ balancer://mycluster/example/
stickysession=JSESSIONID|jsessionid
```

CHAPTER 3. LOAD BALANCING WITH THE JBOSS HTTP CONNECTOR (MOD_PROXY_CLUSTER)

The **mod_proxy_cluster** connector is a reduced-configuration, intelligent load-balancing solution that allows the Apache HTTP Server to connect to back-end JBoss Web Server or JBoss EAP hosts. The **mod_proxy_cluster** module is based on technology that the JBoss **mod_cluster** community project originally developed.

3.1. MOD_PROXY_CLUSTER KEY FEATURES AND COMPONENTS

The **mod_proxy_cluster** module load-balances HTTP requests to JBoss EAP and JBoss Web Server worker nodes. The **mod_proxy_cluster** module uses the Apache HTTP Server as the proxy server.

Key features of mod_proxy_cluster

The **mod_proxy_cluster** connector has several advantages over the **mod_jk** connector:

- When the **mod_proxy_cluster** module is enabled, the **mod_proxy_cluster** Management Protocol (MCMP) is an additional connection between the Tomcat servers and the Apache HTTP Server. The Tomcat servers use MCMP to transmit server-side load figures and lifecycle events back to the Apache HTTP Server, by using a custom set of HTTP methods.
- Dynamic configuration of Apache HTTP Server with **mod_proxy_cluster** allows Tomcat servers that have **mod_proxy_cluster** listeners to join the load-balancing arrangement without the need for manual configuration.
- Tomcat servers perform the load calculations rather than rely on the Apache HTTP Server. This makes load-balancing metrics more accurate than other connectors.
- The **mod_proxy_cluster** connector provides fine-grained application lifecycle control. Each Tomcat server forwards web application context lifecycle events to the Apache HTTP Server. These lifecycle events include informing the Apache HTTP Server to start or stop routing requests for a specific context. This prevents end users from seeing HTTP errors because of unavailable resources.
- You can use Apache JServ Protocol (AJP), Hypertext Transfer Protocol (HTTP), or Hypertext Transfer Protocol Secure (HTTPS) transports with **mod_proxy_cluster**.

Mod_proxy_cluster components

On the proxy server, **mod_proxy_cluster** consists of four Apache modules:

Component	Description
mod_cluster_slotmem.so	The Shared Memory Manager module shares real-time worker node information with multiple Apache HTTP Server processes.
mod_manager.so	The Cluster Manager module receives and acknowledges messages from worker nodes, including node registrations, node load data, and node application life cycle events.

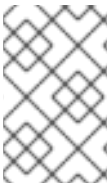
Component	Description
mod_proxy_cluster.so	The Proxy Balancer Module handles request routing to cluster nodes. The Proxy Balancer selects the appropriate destination node based on application location in the cluster, the current state of each of the cluster nodes, and the Session ID (if a request is part of an established session).
mod_advertise.so	The Proxy Advertisement Module broadcasts the existence of the proxy server via UDP multicast messages. The server advertisement messages contain the IP address and port number where the proxy server is listening for responses from worker nodes that want to join the load-balancing cluster.

Additional resources

- **Mod_proxy_cluster** connector modules

3.2. MOD_PROXY_CLUSTER INSTALLATION AND UPGRADE

Red Hat JBoss Core Services (JBCS) and Red Hat Enterprise Linux (RHEL) provide separate distributions of the Apache HTTP Server. The Apache HTTP Server distribution that you install determines whether installation of the **mod_proxy_cluster** connector is automatic or requires a manual step. Depending on your installed distribution of the Apache HTTP Server, the installation path for the **mod_proxy_cluster** modules and configuration files also varies.



NOTE

The JBCS Apache HTTP Server supports the use of **mod_proxy_cluster** on all supported operating systems. The RHEL Apache HTTP Server supports the use of **mod_proxy_cluster** on RHEL 9 only.

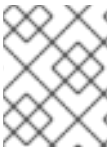
3.2.1. Installation of mod_proxy_cluster when using the JBCS Apache HTTP Server

The Apache HTTP Server part of a JBCS installation automatically installs the **mod_proxy_cluster** module.

You can follow the procedures in the Red Hat JBoss Core Services Apache HTTP Server Installation Guide to install or upgrade to the latest JBCS Apache HTTP Server release for your operating system. For more information, see the **Additional resources** links.

Consider the following guidelines for a **mod_proxy_cluster** installation when using the JBCS Apache HTTP Server:

- The **mod_proxy_cluster.so**, **mod_cluster_slotmem.so**, **mod_manager.so**, and **mod_advertise.so** modules are installed in the ***JBCS_HOME*/httpd/modules** directory.
- The **mod_proxy_cluster.conf.sample** configuration file is located in the ***JBCS_HOME*/httpd/conf.d** directory.
- The **mod_proxy_cluster.conf.sample** file includes a **LoadModule** directive for the **mod_proxy_cluster** module.



NOTE

JBCS_HOME represents the top-level directory for a JBCS installation, which is **/opt/jbcs-httpd24-2.4**.

Additional resources

- Installing the JBCS Apache HTTP Server on RHEL from archive files
- Installing the JBCS Apache HTTP Server on RHEL 7 or RHEL 8 from RPM packages
- Installing the JBCS Apache HTTP Server on Windows Server

3.2.2. Upgrade of mod_proxy_cluster from an earlier JBCS release

The **mod_cluster-native** package that JBCS provided in 2.4.37 and earlier releases is renamed **mod_proxy_cluster** in JBCS 2.4.51 or later. As part of this change, the **mod_cluster.conf** file that was available in 2.4.37 and earlier releases is also renamed **mod_proxy_cluster.conf** in JBCS 2.4.51 or later. JBCS handles the upgrade of your existing **mod_proxy_cluster** configuration in different ways depending on whether you installed JBCS from archive files or RPM packages.

Upgrades of mod_proxy_cluster configuration when installed from RPM packages
If you are upgrading an existing JBCS installation that you installed from RPM packages on RHEL 7 or RHEL 8, consider the following guidelines:

- If you are upgrading from JBCS 2.4.37 or earlier, JBCS retains your existing **mod_cluster.conf** file during the upgrade. In this situation, the upgraded JBCS 2.4.57 deployment includes both your existing **mod_cluster.conf** file and a default **mod_proxy_cluster.conf** file. If you subsequently want to migrate to using **mod_proxy_cluster.conf** instead, you can manually update the default **mod_proxy_cluster.conf** file to suit your setup requirements.
- If you are upgrading from JBCS 2.4.51, JBCS retains your existing **mod_proxy_cluster.conf** file during the upgrade. In this situation, the upgraded JBCS 2.4.57 deployment includes both your existing **mod_proxy_cluster.conf** file and a default **mod_proxy_cluster.conf.rpmnew** file.

Upgrades of mod_proxy_cluster configuration when installed from archive files
If you are upgrading an existing JBCS installation that you installed from archive files, consider the following guidelines:

- If you are upgrading from JBCS 2.4.37 or earlier, you do not need to take any action apart from extracting the 2.4.57 archive files. JBCS 2.4.57 does not include a default **mod_cluster.conf** file, so your existing **mod_cluster.conf** file remains in place during the product upgrade. In this situation, the upgraded JBCS 2.4.57 deployment includes both your existing **mod_cluster.conf** file and a default **mod_proxy_cluster.conf** file. If you subsequently want to migrate to using **mod_proxy_cluster.conf** instead, you can manually update the default **mod_proxy_cluster.conf** file to suit your setup requirements.
- If you are upgrading from JBCS 2.4.51 or an existing release of JBCS 2.4.57, you must first copy your existing **mod_proxy_cluster.conf** file to a temporary location. JBCS 2.4.57 includes a default **mod_proxy_cluster.conf** file, which automatically overwrites your existing **mod_proxy_cluster.conf** file during the product upgrade. After you extract the latest 2.4.57 archive files, you can then copy your backup of the existing **mod_proxy_cluster.conf** file to the correct location to overwrite the default file.

3.2.3. Installing mod_proxy_cluster by using RHEL Application Streams

If you install the RHEL 9 distribution of the Apache HTTP Server from an RPM package by using Application Streams, RHEL does not automatically install the **mod_proxy_cluster** package. In this situation, if you want to use the **mod_proxy_cluster** connector, you must install the **mod_proxy_cluster** package manually.

Prerequisites

- You have installed the Apache HTTP Server on RHEL 9 by using Application Streams.

Procedure

- Enter the following command as the root user:

```
# dnf install mod_proxy_cluster
```

Verification

- To check that the **mod_proxy_cluster** package is successfully installed, enter the following command:

```
# rpm -q mod_proxy_cluster
```

  The preceding command outputs the full name of the installed package, which includes version and platform information.

Consider the following guidelines for a **mod_proxy_cluster** installation when using RHEL Application Streams:

- The **mod_proxy_cluster.so**, **mod_cluster_slotmem.so**, **mod_manager.so**, and **mod_advertise.so** modules are installed in the **/usr/lib64/httpd/modules** directory.
- The **mod_proxy_cluster.conf.sample** configuration file is located in the **/etc/httpd/conf.d** directory.
- The **mod_proxy_cluster.conf.sample** file includes a **LoadModule** directive for the **mod_proxy_cluster** module.

Additional resources

- Application Streams
- Managing software with the DNF tool

3.3. APACHE HTTP SERVER LOAD-BALANCING CONFIGURATION WHEN USING MOD_PROXY_CLUSTER

In the Apache HTTP Server 2.1 and later versions, **mod_proxy_cluster** is configured correctly for the Apache HTTP Server by default. For more information about setting a custom configuration, see Configuring a basic proxy server.

Example configuration file for mod_proxy_cluster

Depending on whether you installed **mod_proxy_cluster** through Red Hat JBoss Core Services (JBCS) or by using Red Hat Enterprise Linux (RHEL) Application Streams, consider the following guidelines:

- JBCS provides an example configuration file for **mod_proxy_cluster** in the ***JBCS_HOME*/httpd/conf.d/** directory.
- RHEL provides an example configuration file for **mod_proxy_cluster** in the **/etc/httpd/conf.d/** directory.

The example configuration file for **mod_proxy_cluster** is named **mod_proxy_cluster.conf.sample**. To use this example instead of creating your own configuration file, you can remove the **.sample** extension, and modify the file content as needed.

NOTE

You can also use the Load Balancer Configuration tool on the Red Hat Customer Portal to generate optimal configuration templates quickly for **mod_proxy_cluster** and Tomcat worker nodes. When you use the Load Balancer Configuration tool for Apache HTTP Server 2.4.57, ensure that you select **2.4.x** as the Apache version, and select **Tomcat/JWS** as the back-end configuration.

Guidelines for using mod_proxy_cluster

Consider the following guidelines for using the **mod_proxy_cluster** connector:

- When you want to use the **mod_proxy_cluster** connector, you must enable the **mod_proxy** module and disable the **mod_proxy_balancer** module.
- If you want **mod_proxy_cluster** to use the Apache JServ Protocol (AJP), you must enable the **proxy_ajp_module**.
- Use AJPSecret **your_secret** to provide the secret for the AJP back end. If **your_secret** does not correspond to the value configured in the back end, the back end sends a **503** error response for any request that is sent through the proxy.

NOTE

Red Hat JBoss Core Services 2.4.57 does not support the tunneling of non-upgraded connections to a back-end websockets server. This means that when you are configuring the **ProxyPass** directive for the **mod_proxy_wstunnel** module, you must ensure that the upgrade parameter is not set to **NONE**. For more information about **mod_proxy_wstunnel**, see the Apache documentation.

3.3.1. Configuring a basic proxy server

You can configure the Apache HTTP Server to function as a proxy server that forwards requests and responses between web clients and back-end web servers. You must configure a proxy server listener to receive connection requests and responses from the back-end worker nodes. When you want to configure a load-balancing proxy server that uses **mod_proxy_cluster**, you must also configure a virtual host for the management channel.

Prerequisites

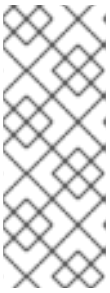
- You have installed the Apache HTTP Server.
- If you installed the RHEL distribution of the Apache HTTP Server by using Application Streams, you have installed **mod_proxy_cluster** manually.

- The port that you specify for the proxy server listener must be open for incoming TCP connections.

Procedure

1. Go to the Apache HTTP Server configuration directory:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.
2. Open the **mod_proxy_cluster.conf** file.
3. To create a **Listen** directive for the proxy server, enter the following line in the **mod_proxy_cluster.conf** file:

```
Listen IP_ADDRESS:PORT_NUMBER
```



NOTE

In the preceding example, replace ***IP_ADDRESS*** with the address of the server network interface that the proxy server uses to communicate with the worker nodes, and replace ***PORT_NUMBER*** with the port that the proxy server listens on.

Ensure that the port is open for incoming TCP connections.

4. To create a virtual host, enter the following details in the **mod_proxy_cluster.conf** file:

```
<VirtualHost IP_ADDRESS:PORT_NUMBER>

  <Directory />
    Require ip IP_ADDRESS
  </Directory>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0

  ManagerBalancerName mycluster
  AdvertiseFrequency 5
  EnableMCPMReceive On

</VirtualHost>
```

NOTE

In the preceding example, replace ***IP_ADDRESS*** and ***PORT_NUMBER*** with the address of the server network interface and port number that you have specified for the **Listen** directive.

This address and port combination is only used for **mod_proxy_cluster** management messages. This address and port combination is not used for general traffic.

For more information about starting the Apache HTTP Server service, see the Red Hat JBoss Core Services Apache HTTP Server Installation Guide.

3.3.1.1. Disabling server advertisement

The proxy server uses UDP multicast to advertise itself. The **AdvertiseFrequency** directive instructs the server to send server advertisement messages every 10 seconds by default. Server advertisement messages contain the ***IP_ADDRESS*** and ***PORT_NUMBER*** that you specify in the **VirtualHost** definition. Worker nodes that are configured to respond to server advertisements use this information to register themselves with the proxy server. If you want to prevent worker nodes from registering with the proxy server, you can optionally disable server advertisement.

NOTE

When UDP multicast is available between the proxy server and the worker nodes, server advertisement adds worker nodes without requiring further configuration on the proxy server. Server advertisement requires only minimal configuration on the worker nodes.

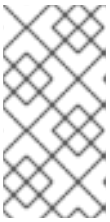
Prerequisites

- You have configured a basic proxy server.

Procedure

1. Go to the Apache HTTP Server configuration directory:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.
2. Open the **mod_proxy_cluster.conf** file.
3. Add the following directive to the **VirtualHost** definition:

```
ServerAdvertise Off
```



NOTE

If server advertisements are disabled, or UDP multicast is not available on the network between the proxy server and the worker nodes, you can configure worker nodes with a static list of proxy servers. In either case, you do not need to configure the proxy server with a list of worker nodes.

Additional resources

- Configuring worker nodes with a static list of proxy servers

3.3.1.2. Logging worker node details

When you configure a load-balancing proxy server that uses **mod_proxy_cluster**, you can optionally configure the Apache HTTP Server to log details of each worker node that handles a request. Logging worker node details can be useful if you need to troubleshoot your load balancer.

Prerequisites

- You have configured a basic proxy server.

Procedure

1. Go to the Apache HTTP Server configuration directory:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.d** directory.
2. Open the **mod_proxy_cluster.conf** file.
3. Add the following details to your Apache HTTP Server **LogFormat** directive(s):

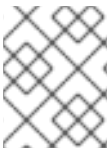
```
%{BALANCER_NAME}e ::
The name of the balancer that served the request.

%{BALANCER_WORKER_NAME}e ::
The name of the worker node that served the request.
```

Additional resources

- Apache HTTP Server documentation about log files

3.3.2. Configuring a JBoss Web Server worker node in mod_proxy_cluster

When you use **mod_proxy_cluster**, you can configure a back-end worker node as a JBoss Web Server Tomcat service that operates in non-clustered mode only. In this situation, **mod_proxy_cluster** can use only one load metric at any specific time when calculating the load-balance factor.



NOTE

JBoss Web Server worker nodes support only a subset of **mod_proxy_cluster** functionality. Full **mod_proxy_cluster** functionality is available with JBoss EAP.

Prerequisites

- You are familiar with the proxy and proxy discovery configuration attributes for **mod_proxy_cluster**.

Procedure

1. To add a listener to JBoss Web Server, in the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file, add the following **Listener** element under the other **Listener** elements:

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="true" stickySession="true" stickySessionForce="false"
stickySessionRemove="true" />
```

2. To give the worker node a unique identity, in the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file, add the **jvmRoute** attribute and value to the **Engine** element:

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="worker01">
```

3. To configure **STATUS MCMP** message frequency, modify the **org.jboss.modcluster.container.catalina.status-frequency** Java system property. For example:

```
-Dorg.jboss.modcluster.container.catalina.status-frequency=6
```

NOTE

JBoss Web Server worker nodes periodically send status messages that contain their current load status to the Apache HTTP Server balancer. The default frequency of these messages is 10 seconds. If you have hundreds of worker nodes, the **STATUS MCMP** messages can increase traffic congestion on your Apache HTTP Server network.

You can configure the **MCMP** message frequency by modifying the **org.jboss.modcluster.container.catalina.status-frequency** Java system property. By default, the property accepts values that are specified in seconds multiplied by 10. For example, setting the property to **1** means 10 seconds. In the preceding example, the property is set to **6**, which means 60 seconds.

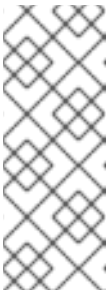
4. Optional: To configure the firewall for proxy server advertisements, complete either of the following steps to open port **23364** for UDP connections on the worker node's firewall:

   - For RHEL:

```
firewall-cmd --permanent --zone=public --add-port=23364/udp
```

   - For Windows Server using PowerShell:

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh
Advfirewall firewall add rule name="UDP Port 23364" dir=in  action=allow protocol=UDP
localport=23364"'
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh
Advfirewall firewall add rule name="UDP Port 23364" dir=out action=allow protocol=UDP
localport=23364"'
```



NOTE

When a proxy server uses **mod_proxy_cluster**, the proxy server can use UDP multicast to advertise itself. Most operating system firewalls block the server advertisement feature by default. To enable server advertisement and receive these multicast messages, you can open port **23364** for UDP connections on the worker node's firewall, as shown in the preceding examples.

3.3.3. Configuring a worker node to operate with a static list of proxy servers

Server advertisement allows worker nodes to discover and register with proxy servers dynamically. If UDP multicast is not available or server advertisement is disabled, you must configure JBoss Web Server worker nodes with a static list of proxy server addresses and ports.

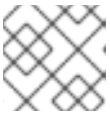
Prerequisites

- You have configured a JBoss Web Server worker node.
- You are familiar with the proxy configuration parameters for Tomcat.

Procedure

1. Open the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.
2. To define a **mod_proxy_cluster** listener and disable dynamic proxy discovery, add or change the **Listener** element for **ModClusterListener**.
   For example:

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true"/>
```



NOTE

Ensure that you set the **advertise** property to **false**.

3. To create a static proxy server list, update the **proxyList** property by adding a comma-separated list of proxies in the following format: ***IP_ADDRESS*:*PORT*,*IP_ADDRESS*:*PORT***
   For example:

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true" proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
```

3.4. MOD_PROXY_CLUSTER CHARACTER LIMITS

The **mod_proxy_cluster** module uses shared memory to keep the nodes description. The shared memory is created at the startup of Apache HTTP Server, and the structure of each item is fixed.

When you define proxy server and worker node properties, ensure that you adhere to the following character limits:

Property	Maximum character limit	Description
Alias length	100 characters	Alias corresponds to the network name of the respective virtual host; the name is defined in the **Host** element.

Property	Maximum character limit	Description
Context length	40 characters	For example, if **myapp.war** is deployed in /**myapp** , /**myapp** is included in the context.
Balancer name length	40 characters	This is the **balancer** in the **<Listener>** element.
JVMRoute string length	80 characters	**JVMRoute** in the **<Engine>** element.
Domain name length	20 characters	This is the **loadBalancingGroup** in the **<Listener>** element.
Hostname length for a node	64 characters	This is hostname address in the **<Connector>** element.
Port length for a node	7 characters	This is the port property in the **<Connector>** element. For example, **8009** is 4 characters.
Scheme length for a node	6 characters	This is the protocol of the connector. Possible values are **http**, **https**, and **ajp**.
Cookie name length	30 characters	This is the header cookie name for the session ID. The default value is **JSESSIONID** based on the **org.apache.catalina.Globals.SESSION_COOKIE_NAME** property.
Path name length	30 characters	This is the parameter name for the session ID. The default value is **JSESSIONID** based on the **org.apache.catalina.Globals.SESSION_PARAMETER_NAME** property.
Session ID length	120 characters	A session ID is in the following type of format: **BE81FAA969BF64C8EC2B6600457EAAAA.node01**

CHAPTER 4. CONFIGURATION EXAMPLE FOR LOAD-BALANCING WITH MOD_PROXY_CLUSTER

You can configure JBCS to use the **mod_proxy_cluster** connector for load-balancing in a Red Hat Enterprise Linux system.

When you want to configure a load-balancing solution that uses **mod_proxy_cluster**, you must perform the following tasks:

1. Set up JBCS as a proxy server.
2. Configure a Tomcat worker node.
3. Define iptables firewall rules.

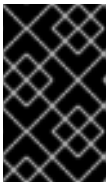
4.1. SETTING UP JBCS AS A PROXY SERVER

When you configure JBCS to use **mod_proxy_cluster**, you must set up JBCS as a proxy server by specifying configuration details in the **mod_proxy_cluster.conf** file.

Procedure

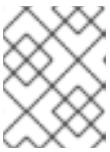
1. Go to the ***JBCS_HOME*/httpd/conf.d/** directory.
2. Create a file named **mod_proxy_cluster.conf**.
3. Enter the following configuration details:

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_proxy_cluster

<IfModule manager_module>
  Listen 6666
  <VirtualHost *:6666>
    <Directory />
      Require ip 127.0.0.1
    </Directory>
    ServerAdvertise on
    EnableMCPMReceive
    <Location /mod_cluster_manager>
      SetHandler mod_cluster-manager
      Require ip 127.0.0.1
   </Location>
  </VirtualHost>
</IfModule>
```



IMPORTANT

As shown in the preceding example, the **mod_proxy_cluster** package requires that you set the **MemManagerFile** directive in the **conf.d** file to **cache/mod_proxy_cluster**.



NOTE

The preceding example shows how to set up JBCS as a proxy server that is listening on **localhost**.

4.2. CONFIGURING A TOMCAT WORKER NODE

When you configure JBCS to use **mod_proxy_cluster**, you must configure a Tomcat worker node by adding a **Listener** element to the **server.xml** file.

Prerequisites

- You have set up JBCS as a proxy server .

Procedure

1. Open the ***JWS_HOME*/tomcat*<VERSION>*/conf/server.xml** file.
2. Add the following **Listener** element:

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="true"/>
```

4.3. DEFINING IPTABLES FIREWALL RULES EXAMPLE

When you configure JBCS to use **mod_proxy_cluster**, you must define firewall rules by using **iptables**.

Prerequisites

- You have configured a Tomcat worker node .

Procedure

- Use **iptables** to define a set of firewall rules.
  For example:

```
/sbin/iptables -I INPUT 5 -p udp -d 224.0.1.0/24 -j ACCEPT -m comment --comment
"mod_proxy_cluster traffic"
/sbin/iptables -I INPUT 6 -p udp -d 224.0.0.0/4 -j ACCEPT -m comment --comment "JBoss
Cluster traffic"
/sbin/iptables -I INPUT 9 -p udp -s 192.168.1.0/24 -j ACCEPT -m comment --comment
"cluster subnet for inter-node communication"
/sbin/iptables -I INPUT 10 -p tcp -s 192.168.1.0/24 -j ACCEPT -m comment --comment
"cluster subnet for inter-node communication"
/etc/init.d/iptables save
```

NOTE

The preceding example shows to define firewall rules for a cluster node on the **192.168.1.0/24** subnet.

APPENDIX A. MOD_PROXY CONNECTOR MODULES

The **mod_proxy** connector comprises a set of standard Apache HTTP Server modules. These modules enable the Apache HTTP Server to act as a proxy/gateway for sending web traffic between web clients and back-end servers over different types of protocols.

This appendix describes the modules that the **mod_proxy** connector uses.

A.1. MOD_PROXY.SO MODULE

The **mod_proxy.so** module is a standard Apache HTTP Server module that enables the server to act as a proxy for data transferred over the AJP (Apache JServe Protocol), FTP, CONNECT (for SSL), and HTTP protocols. The **mod_proxy** module does not require additional configuration. The identifier for the **mod_proxy** module is **proxy_module**.

Additional resources

- Apache Module mod_proxy

A.2. MOD_PROXY_AJP.SO MODULE

The **mod_proxy_ajp.so** module is a standard Apache HTTP Server module that provides support for Apache JServ Protocol (AJP) proxying. By using the **mod_proxy_ajp** module, the Apache HTTP Server acts as an intermediary for sending AJP requests and responses between web clients and back-end servers. AJP is a clear-text protocol that does not support data encryption.

The **mod_proxy** module is also required if you want to use **mod_proxy_ajp**. The identifier for the **mod_proxy_ajp** module is **proxy_ajp_module**.

Additionally, the **secret** property is required when using the Tomcat AJP Connector. You can add the **secret** property to the **ProxyPass** settings by using the following command:

```
ProxyPass /example/ ajp://localhost:8009/example/ secret=YOUR_AJP_SECRET
```

NOTE

If you set a **secret** on a load balancer, all of its members inherit this **secret**.

The **mod_proxy_ajp** module does not provide any configuration directives.

Additional resources

- Apache Module mod_proxy_ajp

A.3. MOD_PROXY_HTTP.SO MODULE

The **mod_proxy_http.so** module is a standard Apache HTTP Server module that provides support for Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) proxying. By using the **mod_proxy_http** module, the Apache HTTP Server acts as an intermediary for forwarding HTTP or HTTPS requests between web clients and back-end servers. The **mod_proxy_http** module supports HTTP/1.1 and earlier versions of the HTTP protocol.

The **mod_proxy** module is also required if you want to use **mod_proxy_http**. The identifier for the **mod_proxy_http** module is **proxy_http_module**.

The **mod_proxy_http** module does not provide any configuration directives. Along with the configuration that controls the behavior of the **mod_proxy** module, the **mod_proxy_http** module uses a series of environment variables that control the behavior of the HTTP protocol provider.

Additional resources

- Apache Module mod_proxy_http

A.4. MOD_PROXY_HTTP2.SO MODULE

The **mod_proxy_http2.so** module is a standard Apache HTTP Server module that provides support for Hypertext Transfer Protocol 2.0 (HTTP/2) proxying. By using the **mod_proxy_http2** module, the Apache HTTP Server acts as an intermediary for forwarding HTTP/2 requests between web clients and back-end servers.

The **mod_proxy_http2** module supports client requests that use HTTP/1.1 or HTTP/2 as a communication protocol. However, the **mod_proxy_http2** module requires that all communication between the Apache HTTP Server and the back-end server uses HTTP/2 only.

For client requests that have the same back-end destination, the Apache HTTP Server reuses the same TCP connection whenever possible. However, even if you want to forward multiple client requests to the same back end, the Apache HTTP Server forwards a separate HTTP/2 proxy request for each HTTP/1.1 client request.

The **mod_proxy** module is also required if you want to use **mod_proxy_http2**. The identifier for the **mod_proxy_http2** module is **proxy_http2_module**.

The **mod_proxy_http2** module does not provide any configuration directives.

NOTE

The **mod_proxy_http2** module is an experimental Apache feature that requires use of the **libnghttp2** library for the core HTTP/2 engine.

Additional resources

- Enabling HTTP/2 for the JBCS Apache HTTP Server
- Apache Module mod_proxy_http2

APPENDIX B. MOD_JK CONNECTOR MODULE

The Apache Tomcat Connector, **mod_jk**, is a web server plug-in that the Apache Tomcat project provides. The Apache HTTP Server can use the **mod_jk** module to load-balance HTTP client requests to back-end servlet containers, while maintaining sticky sessions and communicating over the Apache JServ Protocol (AJP). The **mod_jk** module is included in the Apache HTTP Server part of a JBoss Core Services installation.

The **mod_jk** module requires that you create both a **mod_jk.conf** file and a **workers.properties** file on the Apache HTTP Server host. The **mod_jk.conf** file specifies settings to load and configure the **mod_jk.so** module. The **workers.properties** file specifies back-end worker node details. You must also configure some settings on the [JWSShortName] host to enable **mod_jk** support.

Additional resources

- Load balancing with the Apache Tomcat Connector (**mod_jk**)
- **Workers.properties** file for **mod_jk**

APPENDIX C. MOD_PROXY_CLUSTER CONNECTOR MODULES

The **mod_proxy_cluster** connector is a reduced-configuration, intelligent load-balancing solution based on technology that the JBoss mod_cluster community project originally developed. The **mod_proxy_cluster** connector enables the Apache HTTP Server to act as an advanced load-balancer for forwarding traffic to back-end applications running on JBoss Web Server or JBoss EAP hosts. The **mod_proxy_cluster** connector provides all the functionality of **mod_jk** and additional features such as real-time load-balancing calculations, application life-cycle control, automatic proxy discovery, and multiple protocol support.

This appendix describes the modules that the **mod_proxy_cluster** connector uses.

NOTE

You can configure the **mod_proxy_cluster** connector by using the configurable directives for **mod_proxy**, such as **ProxyIOBufferSize**.

C.1. MOD_MANAGER.SO MODULE AND DIRECTIVES

The cluster manager module, **mod_manager.so**, receives and acknowledges messages from nodes, including worker node registrations, worker node load data, and worker node application life cycle events.

```
LoadModule manager_module modules/mod_manager.so
```

Configurable directives for mod_manager.so

Configurable directives in the **<VirtualHost>** element are as follows:

EnableMCPMReceive

Allows the **VirtualHost** to receive mod_cluster management protocol (MCMP) messages. Add one **EnableMCPMReceive** directive to the Apache HTTP Server configuration to allow **mod_proxy_cluster** to operate correctly. **EnableMCPMReceive** must be added in the **VirtualHost** configuration at the location where **advertise** is configured.

MaxMCMPMaxMessSize

Defines the maximum size of MCMP messages. The default value is calculated from other **Max** directives. The minimum value is **1024**.

AllowDisplay

Toggles the additional display on the **mod_cluster-manager** main page. The default value is **off**, which causes only version information to display on the **mod_cluster-manager** main page.

AllowCmd

Toggles permissions for commands using **mod_cluster-manager** URL. The default value is **on**, which allows commands.

ReduceDisplay

Toggles the reduction of information displayed on the **mod_cluster-manager** page. Reducing the information allows more nodes to display on the page. The default value is **off**, which allows all the available information to display.

MemManagerFile

Defines the location for the files in which mod_manager stores configuration details. mod_manager also uses this location for generated keys for shared memory and lock files. **This must be an**

absolute path name. It is recommended that this path be on a local drive, and not an NFS share. The default value is /**logs**/ .

Maxcontext

The maximum number of contexts that **mod_proxy_cluster** will use. The default value is **100**.

Maxnode

The maximum number of worker nodes that **mod_proxy_cluster** will use. The default value is **20**.

Maxhost

The maximum number of hosts (aliases) that **mod_proxy_cluster** will use. This is also the maximum number of load balancers. The default value is **20**.

Maxsessionid

The maximum number of active session identifiers stored. A session is considered inactive when no information is received from that session for five minutes. This is used for demonstration and debugging purposes only. The default value is **0**, which disables this logic.

ManagerBalancerName

The name of the load balancer to use when the worker node does not provide a load balancer name. The default value is **mycluster**.

PersistSlots

When set to **on**, nodes, aliases, and contexts are persisted in files. The default value is **off**.

CheckNonce

When set to **on**, session identifiers are checked to ensure that they are unique and have not occurred before. The default is **on**.

NOTE

Setting this directive to **off** can leave your server vulnerable to replay attacks.

SetHandler mod_cluster-manager

Defines a handler to display information about worker nodes in the cluster. This is defined in the **Location** element:

```
<Location $LOCATION>
  SetHandler mod_cluster-manager
  Require ip 127.0.0.1
</Location>
```

In this situation, **$LOCATION** was also defined as **mod_cluster_manager**.

Consider the following guidelines when accessing the **$LOCATION** that is defined in the **Location** element in your browser:

- **Transferred** corresponds to the POST data sent to the worker node.
- **Connected** corresponds to the number of requests that had been processed when this status page was requested.
- **Sessions** corresponds to the number of active sessions. This field is not present when **Maxsessionid** is **0**.

C.2. MOD_PROXY_CLUSTER.SO MODULE AND DIRECTIVES

The Proxy Balancer Module, **mod_proxy_cluster.so**, handles the routing of requests to cluster nodes. The Proxy Balancer selects the appropriate node to forward the request to based on the application location in the cluster, the current state of each of the cluster nodes, and the Session ID (if a request is part of an established session).

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
```

Configurable directives for mod_proxy_cluster.so

You can also configure the following directives in the **<VirtualHost>** element to change the load balancing behavior.

CreateBalancers

Defines how load balancers are created in the Apache HTTP Server virtual hosts. The following values are valid in **CreateBalancers**:

- **0**: Create load balancers in all virtual hosts defined in Apache HTTP Server. Remember to configure the load balancers in the **ProxyPass** directive.
- **1**: Do not create balancers. When using this value, you must also define the load balancer name in **ProxyPass** or **ProxyPassMatch**.
- **2**: Create only the main server. This is the default value for **CreateBalancers**.

UseAlias

Defines whether to check that the defined **Alias** corresponds to the **ServerName**. The following values are valid for **UseAlias**:

- **0**: Ignore alias information from worker nodes. This is the default value for **UseAlias**.
- **1**: Verify that the defined alias corresponds to a worker node's server name.

LBstatusRecalTime

Defines the interval in seconds between the proxy calculating the status of a worker node. The default interval is **5** seconds.

ProxyPassMatch; ProxyPass

ProxyPass maps remote servers into the local server namespace. If the local server has an address such as **http://local.com/**, the following **ProxyPass** directive converts a local request for **http://local.com/requested/file1** into a proxy request for **http://worker.local.com/file1**.

```
ProxyPass /requested/ http://worker.local.com/
```

ProxyPassMatch uses regular expressions to match local paths to which the proxied URL should apply.

For either directive, **!** indicates that a specified path is local, and a request for that path should not be routed to a remote server. For example, the following directive specifies that **gif** files should be served locally.

```
ProxyPassMatch ^(/.*\.gif)$ !
```

UseNocanon

Defines whether to forward the original URL path to the back end without modifications.
The default value is **Off**. When the **UseNocanon** directive is set to **Off**, the proxy can forward modified URLs to the back end. However, if the back-end application expects the original URL path that the client requested, the modified URL path can lead to unexpected issues.

When you set the **UseNocanon** directive to **On**, the proxy can forward the original URL path to the back end without any modifications. In this situation, the proxy behavior depends on whether you also define a context and a **ProxyPass** directive for the requested URL in the **mod_proxy_cluster.conf** file. A context is also known as a *virtual host definition*.

Consider the following guidelines when you set the **UseNocanon** directive to **On**:

- If you define a context for the requested URL but you do not define a **ProxyPass** directive for this URL, the proxy uses the **UseNocanon** directive.
- If you define both a context and a **ProxyPass** directive for the requested URL, and the **ProxyPass** directive includes the **nocanon** flag, the proxy uses the **nocanon** flag and ignores the **UseNocanon** directive.
- If you define both a context and a **ProxyPass** directive for the requested URL, and the **ProxyPass** directive excludes the **nocanon** flag, the proxy ignores the **UseNocanon** directive.

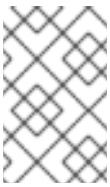
NOTE

If you do not define a context for the requested URL, **mod_proxy_cluster** returns a **404** error.

ResponseStatusCodeOnNoContext

Defines the response status code that the server sends to the client when a **ProxyPass** or **ProxyPassMatch** directive does not have a matching context.
The default value is **404**, which means that the server sends a **Not Found** error response by default.

In JBCS 2.4.51 or earlier, when a **ProxyPass** or **ProxyPassMatch** directive did not have a matching context, the server sent a **503 Service Unavailable** response by default. If you want to preserve the default behavior that was available in earlier releases, set the **ResponseStatusCodeOnNoContext** directive to **503** instead.



NOTE

If you specify a value other than a standard HTTP response code, the server access log shows the specified value but the server sends a **500 Internal Server Error** response to the client.

C.3. MOD_ADVERTISE.SO MODULE AND DIRECTIVES

The Proxy Advertisement Module, **mod_advertise.so**, broadcasts the existence of the proxy server via UDP multicast messages. The server advertisement messages contain the IP address and port number where the proxy is listening for responses from nodes that wish to join the load-balancing cluster.

The **mod_advertise** module must be defined along with the **mod_manager** module in the **VirtualHost** element. In the following example, the identifier for the **mod_advertise** module is **advertise_module**:

```
LoadModule advertise_module modules/mod_advertise.so
```

Configurable directives for mod_advertise.so

The **mod_advertise** module is configurable by using the following directives:

ServerAdvertise

Defines how the advertising mechanism is used.
The default value is **Off**. When set to **Off**, the proxy does not advertise its location.

When set to **On**, the advertising mechanism is used to tell worker nodes to send status information to this proxy. You can also specify a host name and port with the following syntax: **ServerAdvertise On http://*HOSTNAME*:*PORT***/. This is only required when using a name-based virtual host, or when a virtual host is not defined.

AdvertiseGroup

Defines the multicast address to advertise on. The syntax is **AdvertiseGroup *ADDRESS*:*PORT***, where ***ADDRESS*** must correspond to **AdvertiseGroupAddress**, and ***PORT*** must correspond to **AdvertisePort** in your worker nodes.
If your worker node is JBoss EAP-based, and the **-u** switch is used at startup, the default **AdvertiseGroupAddress** is the value passed via the **-u** switch.

The default value is **224.0.1.105:23364**. If a port is not specified, the port defaults to **23364**.

AdvertiseFrequency

The interval (in seconds) between multicast messages advertising the IP address and port. The default value is **10**.

AdvertiseSecurityKey

Defines a string that is used to identify **mod_proxy_cluster** in Apache HTTP Server. By default, this directive is not set and no information is sent.

AdvertiseManagerUrl

Defines the URL that the worker node should use to send information to the proxy server. By default this directive is not set and no information is sent.

AdvertiseBindAddress

Defines the address and port over which to send multicast messages. The syntax is **AdvertiseBindAddress *ADDRESS*:*PORT***. This allows an address to be specified on machines with multiple IP addresses. The default value is **0.0.0.0:23364**.

C.4. MOD_CLUSTER_SLOTMEM.SO MODULE

The **mod_cluster_slotmem.so** module is a shared memory provider for creating and accessing a shared memory segment in which the data sets are organized in "slots".

The **mod_cluster_slotmem** module does not require any configuration directives.

C.5. ADDITIONAL RESOURCES (OR NEXT STEPS)

- Worker node configuration reference for **mod_proxy_cluster**

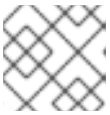
APPENDIX D. WORKERS.PROPERTIES FILE FOR MOD_JK

If you want to use the **mod_jk** connector, you must create a ***JBCS_HOME*/httpd/conf/workers.properties** file on the Apache HTTP Server host to define the back-end worker nodes. The worker nodes are servlet containers that you can map to the **mod_jk** load balancer. The **workers.properties** file specifies the location of the servlet containers and how to load-balance calls across these servlet containers.

This appendix describes the layout and content of the **workers.properties** file.

Depending on the Apache HTTP Server distribution that you are using, the location of the **workers.properties** file varies:

- If you are using the JBCS Apache HTTP Server, the **workers.properties** file is in the ***JBCS_HOME*/httpd/conf.d** directory.
- If you are using the RHEL Apache HTTP Server, the **workers.properties** file is in the **/etc/httpd/conf.d** directory.



NOTE

The RHEL Apache HTTP Server supports the use of **mod_jk** on RHEL 9 only.

D.1. WORKERS.PROPERTIES OVERVIEW

The **workers.properties** file contains a global properties section and a worker properties section.

Global Properties

This section contains directives that apply to all workers.

Worker Properties

This section contains directives that apply to each individual worker.

Each node is defined using the worker properties naming convention. The worker name can only contain lowercase letters, uppercase letters, numbers, and specific special characters (_, /).

The structure of a worker property is **worker.*WORKER_NAME*.*DIRECTIVE***.

worker

The constant prefix for all worker properties.

WORKER_NAME

The arbitrary name given to the worker. For example: **node1**, **node_01**, **Node_1**.

DIRECTIVE

The specific directive required.

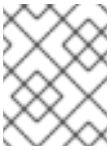
D.2. WORKERS.PROPERTIES DIRECTIVES

The **workers.properties** file directives are divided into global, mandatory, connection, and load-balancing classifications.

Global directives for workers.properties

worker.list

Specifies the list of worker names that **mod_jk** uses. The workers in this list are available to map requests to.



NOTE

A single node configuration which is not managed by a load balancer must be set to **worker.list=*WORKER_NAME***.

Mandatory directives for workers.properties

type

Specifies the type of worker, which determines the directives applicable to the worker. The default value is **ajp13**, which is the preferred worker type to select for communication between the web server and Apache HTTP Server.
Other values include **lb** and **status**.

For detailed information about AJPv13, see the Apache Tomcat Connector - AJP Protocol Reference.

Connection directives for workers.properties

host

The hostname or IP address of the worker. The worker node must support the ajp13 protocol stack. The default value is **localhost**.
You can specify the **port** directive as part of the host directive by appending the port number after the host name or IP address. For example: **worker.node1.host=192.168.2.1:8009** or **worker.node1.host=node1.example.com:8009**.

port

The port number of the remote server instance listening for the defined protocol requests. The default value is **8009**, which is the default listen port for AJPv13 workers.

ping_mode

Specifies the conditions under which connections are probed for their current network health. The probe uses an empty AJPv13 packet for the **CPing**, and expects a **CPong** in return, within a specified timeout.

You specify the conditions by using a combination of the directive flags. The flags are not comma-separated. For example, a correct directive flag set is **worker.node1.ping_mode=CI**.

C (connect)

Specifies the connection is probed once after connecting to the server. You specify the timeout using the **connect_timeout** directive, otherwise the value for **ping_timeout** is used.

P (prepost)

Specifies that the connection is probed before sending each request to the server. You specify the timeout using the **prepost_timeout** directive, otherwise the value for **ping_timeout** is used.

I (interval)

Specifies that the connection is probed during regular internal maintenance cycles. You specify the idle time between each interval using the **connection_ping_interval** directive, otherwise the value for **ping_timeout** is used.

A (all)

The most common setting, which specifies that all directive flags are applied. For information about the ***_timeout** advanced directives, see the Apache Tomcat Connector - Reference Guide.

ping_timeout

Specifies the time to wait for **CPong** answers to a **CPing** connection probe (see **ping_mode**). The default value is **10000** (milliseconds).

Load balancing directives for workers.properties

lbfactor

Specifies the load-balancing factor for an individual worker, and is only specified for a member worker of a load balancer.
This directive defines the relative amount of HTTP request load distributed to the worker compared to other workers in the cluster.

A common example where this directive applies is where you want to differentiate servers with greater processing power than others in the cluster. For example, if you require a worker to take three times the load than other workers, specify **worker.*WORKER_NAME*.lbfactor=3**.

balance_workers

Specifies the worker nodes that the load balancer must manage. The directive can be used multiple times for the same load balancer, and consists of a comma-separated list of worker names as specified in the **workers.properties** file.

sticky_session

Specifies whether requests for workers with SESSION IDs are routed back to the same worker. The default is **0** (false). When set to **1** (true), load balancer persistence is enabled.
For example, if you specify **worker.loadbalancer.sticky_session=0**, each request is load balanced between each node in the cluster. In other words, different requests for the same session can go to different servers based on server load.

If you specify **worker.loadbalancer.sticky_session=1**, each session is persisted (locked) to one server until the session is terminated, providing that server is available.

Additional resources

- Apache Tomcat Connectors - Reference Guide .

APPENDIX E. WORKER NODE CONFIGURATION REFERENCE FOR MOD_PROXY_CLUSTER

E.1. WORKER NODE CONFIGURATION

Configuration values are sent to proxies under the following conditions:

- During server startup
- When a proxy is detected through the advertise mechanism
- During error recovery when a proxy's configuration is reset

Table E.1. Proxy Configuration Values for Tomcat

Value	Default	Description
stickySession	true	Specifies whether subsequent requests for a given session should be routed to the same node, if possible.
stickySessionRemove	false	Specifies whether the Apache HTTP Server proxy should remove session stickiness if the balancer is unable to route a request to the node to which it is stuck. This property is ignored if **stickySession** is **false**.
stickySessionForce	true	Specifies whether the Apache HTTP Server proxy should return an error if the balancer is unable to route a request to the node to which it is stuck. This property is ignored if **stickySession** is **false**.
workerTimeout	-1	Specifies the number of seconds to wait for a worker to become available to handle a request. When all the workers of a balancer are unusable, **mod_proxy_cluster** will retry after a while (**workerTimeout**/**100**) to find an usable worker. A value of **-1** indicates that the Apache HTTP Server will not wait for a worker to be available and will return an error if no workers are available.
maxAttempts	1	Specifies the number of times the Apache HTTP Server proxy will attempt to send a given request to a worker before aborting. The minimum value is **1**: try once before aborting.
flushPackets	false	Specifies whether packet flushing is enabled or disabled.
flushWait	-1	Specifies the time to wait before flushing packets. A value of **-1** means wait forever.

Value	Default	Description
ping	10	Time to wait (in seconds) for a pong answer to a ping.
smax		Specifies the soft maximum idle connection count. The maximum value is determined by the Apache HTTP Server thread configuration (**ThreadsPerChild** or **1**).
ttl	60	Specifies the time (in seconds) idle connections persist, above the **smax** threshold.
nodeTimeout	-1	Specifies the time (in seconds) **mod_proxy_cluster** waits for the back-end server response before returning an error. **mod_proxy_cluster** always uses a**cping**/**cpong** before forwarding a request. The **connectiontimeout** value used by **mod_proxy_cluster** is the ping value.
balancer	mycluster	Specifies the name of the load-balancer.
loadBalancingGroup		Specifies the load balancing among **jvmRoutes** within the same load balancing group. A **loadBalancingGroup** is conceptually equivalent to a mod_jk domain directive.

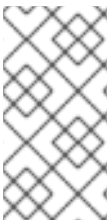
E.2. PROXY AND PROXY DISCOVERY CONFIGURATION ATTRIBUTES FOR MOD_PROXY_CLUSTER

The following tables contain attributes and information about proxy and proxy discovery configuration attributes for **mod_proxy_cluster**.

Table E.2. Proxy discovery configuration attributes for mod_proxy_cluster

Attribute	Property	Default Value
proxy-list	**proxyList**	
proxy-url	**proxyURL**	
advertise	**advertise**	true
advertise-security-key	**advertiseSecurityKey**	
excluded-contexts	**excludedContexts**	

Attribute	Property	Default Value
auto-enable-contexts	**autoEnableContexts**	true
stop-context-timeout	**stopContextTimeout**	10 seconds (in seconds)
socket-timeout	**nodeTimeout**	20 seconds (in milliseconds)



NOTE

When **nodeTimeout** is not defined, the **ProxyTimeout** directive, **Proxy**, is used. If **ProxyTimeout** is not defined, the server timeout (**Timeout**) is used (120 seconds by default in the JBCS httpd.conf). **nodeTimeout**, **ProxyTimeout**, and **Timeout** are set at the socket level.

Table E.3. Proxy configuration attributes for mod_proxy_cluster

Attribute	Property	Default Value
sticky-session	**stickySession**	true
sticky-session-remove	**stickySessionRemove**	false
sticky-session-force	**stickySessionForce**	true
node-timeout	**workerTimeout**	-1
max-attempts	**maxAttempts**	1
flush-packets	**flushPackets**	false
flush-wait	**flushWait**	-1
ping	**ping**	10 (seconds)
smax	**smax**	-1 (uses the default value)
ttl	**ttl**	-1 (uses the default value)
domain	**loadBalancingGroup**	
load-balancing-group	**loadBalancingGroup**	

E.3. LOAD CONFIGURATION FOR TOMCAT

You can configure the following additional properties for load metrics when you want to use **mod_proxy_cluster** with Apache Tomcat.

Table E.4. Load Configuration for Tomcat

Attribute	Default Value	Description
loadMetricClass	**org.jboss.modcluster.load.metric.impl.BusyConnectorsLoadMetric**	The class name of an object that is implementing **org.jboss.load.metric.LoadMetric**
loadMetricCapacity	1	The capacity of the load metric defined via the **loadMetricClass** property
loadHistory	9	The number of historic load values that must be considered in the load balance factor computation
loadDecayFactor	2	The factor by which the historic load values decrease in significance

APPENDIX F. MULTI-PROCESSING MODULES (MPMS)

Red Hat JBoss Core Services includes a variety of multi-processing modules (MPMs). You can use these MPMs to customize how the Apache HTTP Server responds to incoming requests.

NOTE

MPMs are mutually exclusive. You may only enable and use one MPM at any specific time.

F.1. MPMS OVERVIEW

Multi-processing modules (MPMs) are available for both Red Hat Enterprise Linux (RHEL) and Windows Server. On RHEL, the default MPM varies depending on the operating system version.

MPMs for RHEL

prefork

The **prefork** MPM implements a non-threaded, pre-forking web server. The **prefork** MPM uses a single control process, which launches child processes that listen for and service incoming connections. A single process handles a specific request, which ensures that each request is isolated and does not affect any other requests.

NOTE

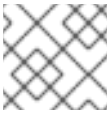
The **prefork** MPM is the default MPM on RHEL 7.

worker

The **worker** MPM implements a hybrid multi-process, multi-threaded server. Each child process creates a fixed number of server threads, which allows the server to handle a large number of requests with fewer system resources.

event

The **event** MPM is based on the **worker** MPM. The **event** MPM allows additional requests to be served simultaneously by delegating some processing work to the listener threads, which frees up the worker threads to serve new requests.



NOTE

The **event** MPM is the default MPM on RHEL versions 8 and 9.

MPMs for Microsoft Windows

winnt

The **winnt** MPM is the only MPM that is available for Windows systems. The **winnt** MPM uses a single control process, which launches another process that creates threads for incoming requests.

F.2. SWITCHING THE MPM

The server selects the MPM based on the **LoadModule** directives in the **00-mpm.conf** file on the Apache HTTP Server host. You can select a specific MPM by removing the comment character (**#**) from the **LoadModule** directive for that MPM in the **00-mpm.conf** file.

Depending on the Apache HTTP Server distribution that you are using, the location of the **00-mpm.conf** file varies:

- If you are using the JBCS Apache HTTP Server, the **00-mpm.conf** file is in the ***JBCS_HOME*/httpd/conf.modules.d** directory.
- If you are using the RHEL Apache HTTP Server, the **00-mpm.conf** file is in the **/etc/httpd/conf.modules.d** directory.

Depending on the operating system version that you are using, consider the following guidelines:

- On RHEL versions 8 and 9, the **event** MPM is selected by default. For example:

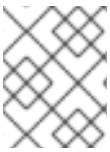
```
# event MPM: A variant of the worker MPM with the goal of consuming
# threads only for connections with active processing
# See: http://httpd.apache.org/docs/2.4/mod/event.html
#
LoadModule mpm_event_module modules/mod_mpm_event.so
```

  The **event** MPM is multi-threaded and designed to provide optimized performance. If you are using RHEL version 8 or 9, switching to another MPM such as **prefork** might lead to performance issues.

- On RHEL 7, the **prefork** MPM is selected by default. For example:

```
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
```

  If you are using RHEL 7, consider switching to another MPM such as **worker** or **event** to avoid possible performance issues.



NOTE

For illustrative purposes, the following procedure describes how to switch from the **prefork** MPM to the **worker** MPM.

Procedure

1. Go to the directory that contains the **00-mpm.conf** file:
   - If you are using the JBCS Apache HTTP Server, go to the ***JBCS_HOME*/httpd/conf.modules.d** directory.
   - If you are using the RHEL Apache HTTP Server, go to the **/etc/httpd/conf.modules.d** directory.
2. Edit the **00-mpm.conf** to add a comment (**#**) character to the **LoadModule** directive for the **prefork** MPM.
   For example:

```
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
```

3. In the same **00-mpm.conf** file, remove the comment (**#**) character from the **LoadModule** directive for the MPM that you want to switch to. These lines are located immediately below the **prefork** MPM.
   For example, to load the **worker** MPM, remove the comment (**#**) character from the **LoadModule** directive for the **worker** MPM:

```
# worker MPM: Multi-Processing Module implementing a hybrid
# multi-threaded multi-process web server
# See: http://httpd.apache.org/docs/2.4/mod/worker.html
LoadModule mpm_worker_module modules/mod_mpm_worker.so
```

Verification

- To verify that the MPM is configured correctly, enter the following command:

```
$ sbin/apachectl -V
```

The preceding command displays the current MPM.

For example:

```
Server MPM:     worker
```

F.3. MPM PERFORMANCE SETTINGS

For each type of MPM, you can configure various settings to optimize the MPM performance.

Types of MPM performance settings

MPM performance settings specify the following types of criteria:

- Initial number of server processes to create at startup
- Minimum and maximum number of idle threads or server processes
- Maximum number of threads or server processes available to handle requests
- Maximum number of requests an individual server process can handle
- Number of threads each server process creates (**worker** and **event** MPMs only)
- Upper limit for the maximum number of server processes that can start during the lifetime of the server (**prefork** MPM only)

Configuration file for MPM performance settings

In JBCS 2.4.51 or later, you can configure MPM performance settings in the **mpm.conf** file. Depending on the Apache HTTP Server distribution that you are using, the location of the **mpm.conf** file varies:

- If you are using the JBCS Apache HTTP Server, the **mpm.conf** file is in the ***JBCS_HOME*/httpd/conf.d** directory.

- If you are using the RHEL Apache HTTP Server, the **mpm.conf** file is in the **/etc/httpd/conf.d** directory.

IMPORTANT

In JBCS 2.4.37 or earlier releases, the **conf.modules.d/00-mpm.conf** file contained the MPM performance settings. From JBCS 2.4.57 onward, the **conf.d/mpm.conf** file contains these settings.

If you are upgrading from JBCS 2.4.37 or earlier, ensure that you configure the **conf.d/mpm.conf** file for your upgraded 2.4.57 installation to match any customized settings that you previously configured in **conf.modules.d/00-mpm.conf**. Otherwise, your upgraded JBCS 2.4.57 installation automatically uses the default settings in the **conf.d/mpm.conf** file, which might lead to unexpected performance issues.

For more information about the available performance settings and associated default values, see the **conf.d/mpm.conf** file in your Apache HTTP Server installation.

Additional resources

- Apache MPM Common Directives